SW Module and Unit Design

Customer

Project

**Revision History**

| **Version** | **Date** | **Change Description / Reason** | **Author** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

1. Introduction 4

1.1 Definitions, Glossary 4

2. Module Design 4

2.1 Description 4

2.2 External Interface 4

2.2.1 Function Interface 5

2.2.2 Message Interface 6

2.2.3 Event Interface 6

2.2.4 Timer Interface 6

2.2.5 Global Variables 7

2.2.6 Module class 7

2.2.7 ASIL-Classification 7

2.2.8 Module specific risks 7

3. Unit Design 7

3.1 Static DoAnyThing (u16 In, s8 & Out) 8

3.1.1 Description 8

3.1.2 Finite State Diagram 8

3.1.3 State/Event Matrix and Action Table 8

3.1.4 Decision Table 9

3.1.5 Flow Chart 9

3.2 Static DoSomethingElse (Void) 11

3.2.1 Description 11

# Introduction

This document applies to project <project name> and is binding for all phases of the project. The document forms the basis for the various template documents which will be drawn up by the process department.

## Definitions, Glossary

If and when required, specify own, project-specific definitions and/or abbreviations.

<Reference to low-level design>

<Other references>

# Module Design

Module API (global functions)

The following needs to be described for the global functions.

**Functionality**: Globally describe what the function is for.

**Return value**: Describe the meaning and the range of the return value.

**Input parameters**: Describe the parameters (content and range) the values of which are and read only.

**Output parameters**: Describe the parameters (content and range) the values of which are written only.

**Parameters which are read and** written: Describe the parameters (content and range) the values of which are both written and read.



## Description

Describe the module’s function. What does the module do, what must be observed etc.

If external interfaces ae already described in another document, the referenced document may be included in chapter 1.1 “Definitions, Glossary“. In this case, it is sufficient just to list the external interfaces used by this module unit.

If not, the external interfaces should be described here.

## External Interface

The following needs to be described for the external interfaces.

**Type**: Describe what type of interface it is: Function call, Message, Timer etc

**Return value**: Describe the meaning and the range of the return value.

**Input parameters**: Describe the parameters (content and range) the values of which are and read only.

**Output parameters**: Describe the parameters (content and range) the values of which are written only.

**Parameters which are read and written**: Describe the parameters (content and range) the values of which are both written and read.

Special attention needs to be given to the operating system interface:

* Operating system interface
* Messages
* Events
* Timers
* Global variables (if unavoidable)

In the following sections, an example is provided for each of the possibilities. If several interfaces of the same type are contained in the module, then the tables may be extended.



### Function Interface

The global C functions within a module are interface and unit at the same time. The prototype of the function is described in this chapter, the function in chapter 1.1.

The following needs to be described:

* Name of the function with parameters and return value (prototype)
* Short description of the purpose
* Description of the parameters
* Value range of the parameters
* Input, output or bidirectional parameters

|  |  |
| --- | --- |
| **Prototype:** | ReturnValue MOD\_ActivateModule (u16 BxY\_u16, s32 & RtFM\_s32) |
| **Description:** | This function activates the module by writing the value to input at port BxY and returning the value of port RtFM in the Output variable. ReturnValue is used to indicate whether the activation was successful. |
| **Parameter name** | **Access type** | **Range** | **Resolution** | **Description** |
| BxY | Read only | Integer between 12 and 18 | Hz\*10-3 | Value to be written to port BxY and indicating the flashing frequency. |
| RtFM | Write only | Integer between -120 and 50000 | °C\*10-1 | Value read from port RtFM. The value indicates the temperature. |
| ReturnValue | Return value of the function | FUNC\_OK  FUNC\_NOK | --- | If the function is terminated normally, then FUNC\_OK is returned. If an error has occurred, then FUNC\_NOK is returned. |

Table 1: Function interface

A table like Table 1 needs to be drawn up for all global C functions.

### Message Interface

The message interface is described as shown below.

|  |  |
| --- | --- |
| **Prototype:** | VRL\_MSG\_SQ\_DOOR\_OPEN |
| **Description:** | This message signals that a door has opened. The message contains as parameter which door has opened. |
| **Write/read:** | Write |
| **Parameter name** | **Range** | **Resolution** | **Description** |
| VRL\_MSG\_PAR\_TUER\_ID | VL | VR | HL | HR | HECK | ----- | Indicates which door is open.  VL ::= front left  VR ::= front right  HL ::= rear left  HR ::= rear right  HECK ::= rear door |

Table 2: Message interface

### Event Interface

The following needs to be documented for an event interface:

* Event name
* Event description
* Is the event sent or received by the module?

|  |  |
| --- | --- |
| **Name:** | VRL\_EV\_KL\_SIG\_CHANGE |
| **Description:** | This event occurs if the KL signal changes. |
| **Send/receive** | Sent |

Table 3: Event interface

### Timer Interface

The following needs to be documented for a timer:

* Timer name
* Timer description
* Timer duration
* Continuously or once

|  |  |
| --- | --- |
| **Name:** | VRL\_TI\_SQ\_DOOR\_OPEN |
| **Description:** | This timer starts when a door is opened. |
| **Duration:** | 3 minutes | Once |

Table 4: Timer interface

### Global Variables

The following needs to be documented for a global variable:

* Variable name
* Variable description
* Variable type
* Is the variable written or read?

|  |  |
| --- | --- |
| **Name:** | VRL\_SwitchLightOn |
| **Description:** | If this variable is True, then the light is switched on, otherwise the light is off. |
| **Type:** | Boolean |
| **Read/write:** | Read only |

Table 5: Global variables

### Module class

Denote the module class (low, medium, high) here.

### ASIL-Classification

If applicable denote the ASIL classification of the module here.

### Module specific risks

Denote here the module specific risks. If they are risks that also have an impact on the overall SW also follow the risk management process.

# Unit Design

The units of the module are described in the following sections. A description must be provided for all units contained in the module. There are several ways to describe a unit, namely:

* Finite state diagram
* State/event matrix with action table
* Decision table
* Flow chart
* Text (for small units)

Which method of description is used is basically up to the developer. In principle, one can say that finite automatons can best be described in a finite state diagram or a state/event matrix. If there is a one-to-one relationship in a unit between input and output variable, then a decision table may be used.

Examples of these diagrams are shown in the next sections.

If the diagrams or tables are already described in other documents, then a reference to those documents suffices here (e.g. Innovator).



## Static DoAnyThing (u16 In, s8 & Out)



### Description

Short description of the function.

### Finite State Diagram

In a finite state diagram, a finite automaton is modelled. The various states in which the system may be are represented by rectangles with the names of the states in them.

The transitions from one state to another are represented by arrows. The event triggering a transition is written next to the arrows. If an action needs to be carried out during the transition, then that action is written next to the transition event, separated by a slash. Also see.

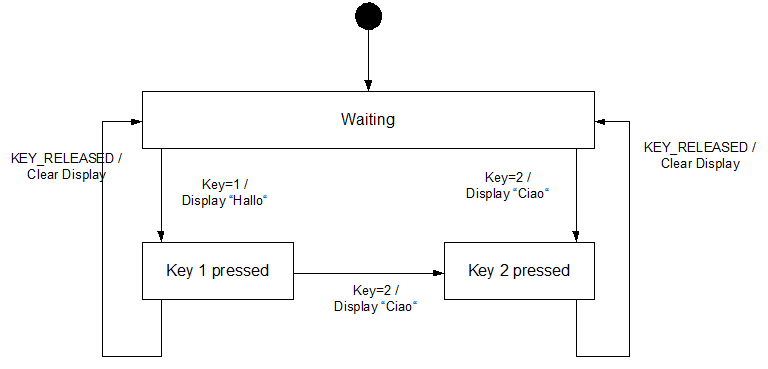


Fig. 1: Finite state diagram

### State/Event Matrix and Action Table

A state/event matrix (SEM) describes to which state the system will change if a certain event occurs. The black cells indicate that the event cannot occur in that state. An action table belongs to an SEM. An action table contains those actions which need to be carried out during a state transition. The actions in the table are consecutively numbered from A1 to An. The action belonging to a transition is indicated behind the slash in the SEM. If “-“ is written behind the slash, e.g. Z5/-, this means that no action needs to be carried out. For instance, if the system is in state Z1 and event Key=1 occurs, then the system will afterwards be in state Z2, as can be derived from the table. Additionally, action A1 is carried out simultaneously during the transition. The action table shows that A1 means that “Hello“ is written to the display.

The state/event matrix is used to be able to draw the finite state diagram. In this way, it can be checked whether all states and events are covered.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Z1 (Waiting) | Z2 (Key 1 pressed) | Z3 (Key 2 pressed) |
| Key=1 | **Z2/A1** |  |  |
| Key=2 | **Z3/A2** | **Z3/A2** |  |
| Key\_Released |  | **Z1/A3** | **Z1/A3** |

Table 6: State/event matrix

|  |  |
| --- | --- |
| **Action number** | **Action** |
| A1 | Write “Hello“ to the display |
| A2 | Write “Ciao“ to the display |
| A3 | Delete the display |

Table 7: Action table

In order to preserve clarity, not more than 20 events must take place in a state diagram. If more than 20 events occur, then the unit needs to be subdivided.

### Decision Table

In a stateless module, a decision table may be used to describe the module. A decision table describes the output variables with the help of the values of the input variables.

|  |  |
| --- | --- |
| *Input* | *Output* |
| *State* | *Key* | *Display* | *State* |
| WAITING | 1 | ”Hello“ | KEY\_1\_PRESSED |
| WAITING | 2 | ”Ciao“ | KEY\_2\_PRESSED |
| KEY\_1\_PRESSED | 2 | ”Ciao“ | KEY\_2\_PRESSED |
| KEY\_1\_PRESSED | RELEASED | CLEAR | WAITING |
| KEY\_2\_PRESSED | RELEASED | CLEAR | WAITING |

Table 8: Decision table

### Flow Chart

A flow chart describes the program sequence by means of arrows which are followed from top to bottom. The appropriate operations and decisions are represented by boxes of various shapes.

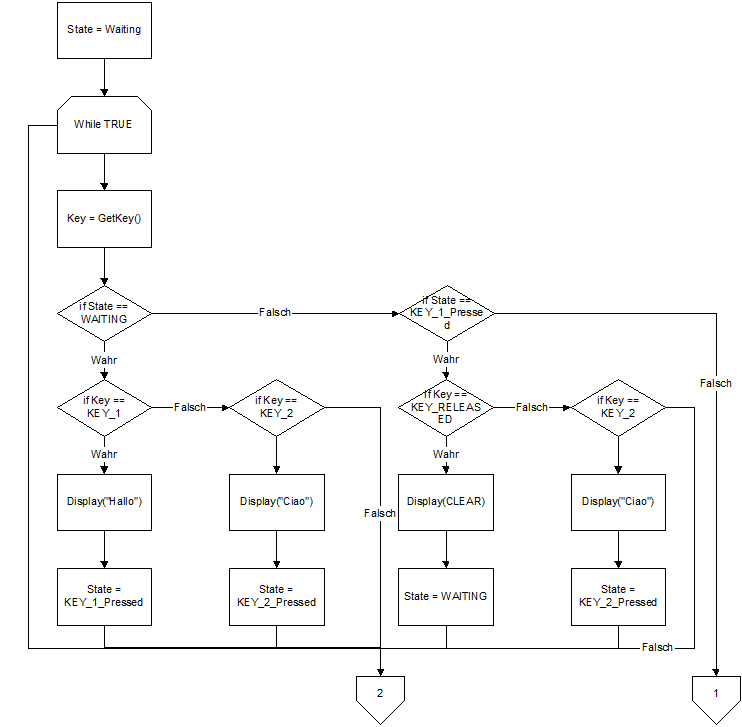


Fig. 3: Flow chart

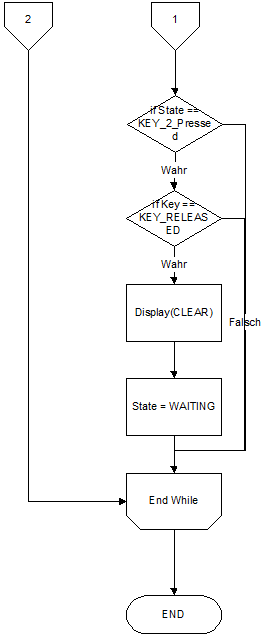


Fig. 4: Flow chart - continued

## Static DoSomethingElse (Void)



### Description

And so on.